IN THE UNITED STATES PATENT AND TRADEMARK OFFICE

APPLICATION FOR LETTERS PATENT

MANAGING XML DOCUMENTS CONTAINING HIERARCHICAL DATABASE INFORMATION

Inventors:
Joseph J. Brychell, III
Xavier Bocken
Robert Jervis
Kamaljit S. Bath
Arungundram Narendran
Mikhail Vassiliev
Danny van Velzen
Nora S. Selim
Hagen Green

ATTORNEY'S DOCKET NO. MS1-1612US
CLIENT'S DOCKET NO. 304177.01

EV395542325

MANAGING XML DOCUMENTS CONTAINING HIERARCHICAL DATABASE INFORMATION

TECHNICAL FIELD

[0001] The described subject matter relates to computing, and more particularly to managing XML documents containing hierarchical database information.

BACKGROUND

[0002] Extensible markup language (XML) is increasingly becoming the preferred format for transferring data. XML is a tag-based hierarchical language that provides the ability to represent data in diverse formats and contexts. For example, XML can be used to represent data spanning the spectrum from semi-structured data (such as one would find in a word-processing document) to generally structured data. XML is well-suited for many types of communication including business-to-business and client-to-server communication.

[0003] Data represented in XML is often created and retained in electronic documents, such as electronic forms. The structure of an electronic form that is written in XML typically is governed by an XML schema (XSD) and this structure can be altered by applying an eXtensible Style-sheet Language Transformation (XSLT) file on the form. For more information on XML, XSLT, and XSD (schemas), the reader is referred to the following documents which are the work of, and available from the W3C (World Wide Web consortium): XML Schema Part

2: Datatypes; XML Schema Part 1: Structures; and XSL Transformations (XSLT) Version 1.0; and XML 1.0 Second Edition Specification.

[0004] Databases are computerized information storage and retrieval systems. A Relational Database Management System (RDBMS) is a database management system (DBMS) which uses relational techniques for storing and retrieving data. RDBMS software using a Structured Query Language (SQL) interface is well known in the art. The SQL interface has evolved into a standard language for RDBMS software and has been adopted as such by both the American National Standards Organization (ANSI) and the International Standards Organization (ISO). The information stored in a RDBMS is typically hierarchical in nature. Parent-child relationships are defined through integrity constraints across multiple relational tables. But it frequently makes sense to present this information to humans (or other applications) as hierarchical information, which can be done using XML.

[0005] When hierarchical database information is persisted as a structured XML document, a unique set of challenges are created in translating the changes to the XML document back into the database. For example, while XML data is by nature hierarchical, typically the only relationship between a parent and a child is the fact that the child is contained within the parent. By contrast, in a database both parent and child will contain common data kept in sync through referential integrity.

[0006] When XML is used as a temporary persistence format for data retrieved from a hierarchical database to be consumed by a wide variety of applications (*e.g.*, web services, internet application, thin client applications), it is important that the retrieved information can be returned to its original storage format (*e.g.*, a database) properly reflecting the changes made by the application(s) that manipulated the data.

SUMMARY

[0007] Described herein are systems and methods for managing XML documents containing hierarchical database information. The systems and methods permit hierarchical database information to be downloaded into an XML document for presentation to a user, or to an application, for editing. Changes to the data made during the editing process are tracked, and the data may be uploaded back to the database.

BRIEF DESCRIPTION OF THE DRAWINGS

[0008] Figs. 1A-1B are schematic illustrations of a mapping between hierarchical data such as, *e.g.*, data stored in a relation database and an XML schema.

[0009] Fig. 2 is a schematic illustration of an exemplary system for managing structured XML documents containing hierarchical data.

[0010] Fig. 3 is a flowchart illustrating operations in an exemplary method for managing XML documents containing hierarchical database information.

[0011] Fig. 4 is a flowchart illustrating operations in an exemplary method for preparing XML data retrieved from the database and annotating the XML data.

[0012] Fig. 5 is a flowchart illustrating operations in an exemplary method for modifying the XML annotations during editing operations to the XML data.

[0013] Fig. 6 is a flowchart illustrating operations in an exemplary method for maintaining the validity of the edited XML data during editing operations.

[0014] Fig. 7 is a flowchart illustrating operations in an exemplary method for identifying changes made to the edited XML data.

[0015] Fig. 8 is a schematic illustration of an exemplary computer system.

DETAILED DESCRIPTION

[0016] The subject matter described herein facilitates management of data that is stored in a hierarchical format such as, *e.g.*, a relational database, when the data is retrieved and manipulated using a schema-driven format such as, *e.g.*, XML. In one implementation this is accomplished by generating a non-editable copy of the XML data retrieved from the database and annotating each logical unit in the original XML data or the copy of the XML data in a manner that uniquely identifies each logical unit and relates it to its corresponding item in the generated copy. For example, each XML node may be assigned a unique numerical or string

identifier or a value representing the items original ordinal position. As the data is manipulated, algorithms may be implemented to use the annotations to track changes to the XML data and to ensure that the manipulated XML data complies with one or more required data formats. When the XML data is ready to be transferred back to the database(s) from which it was obtained, a series of operations are implemented to validate the data and to determine the nature of operation to be performed to restore the data to the database(s).

[0017] This document provides a description of representing XML data retrieved from a hierarchical data source such as, *e.g.*, a database. This document further provides a description of a computer-based system for managing XML documents containing hierarchical database information, exemplary methods performed by such a system, and an exemplary computer on which the system may be implemented. The methods described herein may be embodied as logic instructions on a computer-readable medium implemented in any suitable software or firmware, or may be reduced to hardware. When executed on a processor, the logic instructions cause a general purpose computing device to be programmed as a special-purpose machine that implements the described methods. The processor, when configured by the logic instructions to execute the methods recited herein, constitutes structure for performing the described methods.

Representation of Hierarchical Data Using XML

[0018] **Figs. 1A-1B** are schematic illustrations of the mapping between hierarchical data such as, *e.g.*, data stored in a relation database and an XML schema. Referring to Fig. 1A, a conventional relational database may be considered to store data in the form of one or more tables such as customers table 100 and orders table 130. Customers table 100 may include a plurality of fields represented as columns such as, *e.g.*, CustomerID 112, CompanyName 114, ContactName 116, and Title 118. Similarly, orders table 130 may include a plurality of fields represented as columns such as, *e.g.*, CustomerID 132, OrderID 134, OrderDate 136, and DeliveryDate 138. Customers table 100 and orders table 130 may be related by two or more fields (i.e., one in each table). In Fig. 1, customers table 100 and orders table 130 are related by CustomerID field 112 and CustomerID field 132.

[0019] Relational data from may be extracted or viewed by way of a hierarchical self-describing format such as extensible markup language (XML). For example, SQL Server 2000 developed by MICROSOFT CORPORATION provides extensive database programming capabilities built on Web standards. XML and Internet standard support in that product provides the ability to store and retrieve data in XML format easily with built-in stored procedures. The data extracted from a relational database is converted to an XML document as defined by way of an XML schema.

[0020] An XML Schema is an XML vocabulary for describing XML instance documents. The term "instance" is used because a schema describes a class of documents, of which there can be many different instances.

[0021] The elements used in a schema definition come from the http://www.w3.org/2001/XMLSchema namespace. Schema definitions must have a root xsd:schema element. There are a variety of elements that may be nested within xsd:schema including, but not limited to, xsd:element, xsd:attribute, and xsd:complexType, all of which are well known in the art. A schema definition can be processed with standard XML tools and services such as DOM, SAX, XPath, and XSLT.

[0022] Definitions placed within the xsd:schema element are automatically associated with the namespace specified in the targetNamespace attribute. The namespace identifier is the key that links XML documents to the corresponding Schema definition. To take advantage of the schema while processing an element, the processor needs to locate the correct schema definition. How schema processors locate the schema definition for a particular namespace is not defined by a standard specification. Most processors, however, allow an in-memory cache loading of schemas that it will use while processing documents.

[0023] XML Schema also provides the schemaLocation attribute to provide a hint in the instance document as to the whereabouts of the required schema definitions. The schemaLocation attribute is in the

http://www.w3.org/2001/XMLSchema-instance namespace, which was set aside specifically for attributes that are only used in instance documents.

[0024] **Fig. 1B** is a schematic representation of the mapping between data in relational tables 100 and 130 to an XML schema. The XML schema includes a root node 150, and customers nodes 150a, 150b, that implicate entries in the customers table 100. The child nodes of the customers nodes 150a, 150b may represent other columns in the customers table 100 or may represent data in a table related to the customers table 100. In the implementation depicted in Fig. 1B the customers nodes include CompanyName child nodes 156a, 156b corresponding to entries in the ComanyName column 114 in customers table 100, ContactName nodes 158a, 158b corresponding to entries in the ContactName column 116 in customers table 100, and orders child nodes 160a, 160b corresponding to orders in the order table 130.

[0025] The orders nodes 160a, 160b may have child nodes that correspond to columns in the order table 130. In the embodiment depicted in Fig. 1B the orders nodes 160a, 160b include CustomerID nodes 162a, 162b that correspond to entries in the OrderID column 134 of the orders table 130, OrderID nodes 164a, 164b that correspond to entries in the OrderID column of orders table 130, OrderDate nodes 166a, 166b that correspond to entries in the OrderDate column 136 of orders table 130, and DeliveryDate nodes 168a, 168b that correspond to entries in the DeliveryDate column 138 of orders table 130.

An Exemplary System Architecture

[0026] **Fig. 2** is a schematic depiction of an exemplary system 200 which may be used to manage XML documents containing hierarchical data. This system 200 includes a display 202 having a screen 204, one or more user-input devices 206, and a computer 208.

[0027] The user-input devices 206 can include any device allowing a computer to receive a developer's input, such as a keyboard 210, other device(s) 212, and a mouse 214. The other device(s) 212 can include a touch screen, a voice-activated input device, a track ball, and any other device that allows the system 200 to receive input from a developer. The computer 208 includes a processing unit 216 and random access memory and/or read-only memory 218. Memory 218 includes an operating system 220 for managing operations of computer 208 and one or more application programs, such as database module 220 that interfaces with a database 240 or other source of hierarchical data., an XML processing module 226, an XML editing module 228, an XML conversion module 230, and an XML re-annotation module 232. Memory 218 may further include an XML data store 242. XML data may be persisted (typically temporarily) in XML data store 242. The computer 208 communicates with a user and/or a developer through the screen 204 and the user-input devices 206.

[0028] The system 200 enables a user to retrieve hierarchical data from the database 240, display the data on display 202 using XML formatting techniques, manipulate the retrieved data in its XML format, and to return the manipulated XML data back to database 240. System 200 also monitors changes to the data to provide a measure of data integrity. The operation of various modules 224-232 is explained in detail below.

Exemplary Operations

[0029] **Fig. 3** is a flowchart illustrating operations in an exemplary method for managing XML documents containing hierarchical database information. Referring to Fig. 3, at operation 300 data is retrieved from a database. In an exemplary implementation the database may be a relational database such as database 240, and data may be retrieved by database module 224 using an XML-capable SQL service, as described above. At operation 400 the XML data is annotated and prepared for manipulation by the user(s) of the data, *e.g.*, by XML processing module 226. At operation 500 the annotations are modified during editing operations to the XML data, *e.g.*, by XML editing module 228. At operation 600 the validity of the edited data is maintained during editing operations, *e.g.*, by XML editing module 228. At operation 700 changes to the XML data are identified and the database(s) are updated to reflect changes made to the XML data, *e.g.*, by XML conversion module 230. At operation 800 the XML

data is re-annotated for continued editing, if desired, *e.g.*, by XML re-annotation module 232. Operations 400-800 are explained in greater detail below, with reference to Figs. 4-7.

[0030] **Fig. 4** is a flowchart illustrating operations in an exemplary method for preparing the XML data retrieved from the database and annotating the XML data and annotating the XML data. In an exemplary implementation the operations of Fig. 4 may be implemented by the XML processing module 226. At operation 410 the XML data is received from the database, *e.g.*, as a result of an SQL query. At operation 415 a copy of the received XML data is generated. To distinguish the copy from the original retrieved XML data, the copy will be referred to herein as the before copy.

[0031] At operation 420 the original XML data is annotated in a manner that uniquely identifies each node in the XML data. Optionally, the before copy of the XML data may also be annotated. In an exemplary implementation each node in the XML data is annotated with a name and an identifier that identifies the position of the corresponding node in the before copy of the data. The annotated identifiers may be implemented as a numeric sequence, *e.g.*, a sequential numeric listing, or a predetermined character sequence.

[0032] The identifier may be added to the XML data using a node attribute on each node to which an identifier is assigned. To avoid conflicts between the annotated identifier and other XML data the identifier attribute is assigned a

namespace that is unique to the update algorithm. In addition, to avoid conflicts between identifiers in multiple XML documents the identifier attribute name is unique across multiple XML documents. In an exemplary implementation the attribute name may be randomly generated each time the XML data is prepared and annotated.

[0033] After the XML data is prepared and annotated it may be displayed as an XML document (*e.g.*, on display 202) and modified, either automatically or by a user of system. During modification of the XML document data may be inserted into the XML document in several different ways. By way of example, data may be entered manually, moved (*i.e.*, cut and pasted) from another part of the XML document, or another XML document, or copied and pasted from another part of the XML document, or another XML document.

[0034] **Fig. 5** is a flowchart illustrating operations in an exemplary method for modifying the XML annotations during editing operations to the XML data. In an exemplary implementation the operations of Fig. 5 may be performed by XML editing module 228. At operation 510 a new node is received in the XML data, *e.g.*, as a result of a copy operation a move operation, or an insert operation. At operation 515 it is determined whether the new node includes an annotated identifier. If the new node does not include an annotated identifier, then the new node is likely to be a newly inserted node, and control is returned to the calling routine at operation 535.

[0035] By contrast, if at operation 515 there is an annotated identifier in the new node, then control passes to operation 520, and the XML document is searched for the annotated identifier. The search is performed based on the update algorithm namespace, the unique attribute name and the annotated identifier. If at operation 525 the annotated identifier exists in the XML document, then the new node is likely a copy of an existing node in the XML document. Accordingly, the annotated identifier is removed from the new node at operation 530.

[0036] By contrast, if the annotated identifier does not exist elsewhere in the XML document, then the new node likely represents a move operation of the data from elsewhere in the XML document. Accordingly, control can be returned to the calling routine at operation 535.

[0037] **Fig. 6** is a flowchart illustrating operations in an exemplary method for maintaining the validity of the edited XML data. The operations of Fig 6 may be performed contemporaneously with the operations of Fig. 5, and may be implemented by XML editing module 228. At operation 610 a new node is received in the XML data, as described above in connection with operation 510. At operation 615 the data changes are validated against the XML schema. If, at operation 620 the data changes are inconsistent with the XML schema, then control passes to operation 635 and the changes are rejected. By contrast, if the data changes are consistent with the XML schema, then control passes to operation 625 and the data changes are validated against database constraints. By way of

example, the data may be validated to ensure that PrimaryKey – SecondaryKey constraints between rows in the database are not violated. If the data changes violate one or more database constraints, then control passes to operation 635 and the changes are rejected.

[0038] In an exemplary implementation, constraints are checked in a delayed manner before submit on rows with empty values for the Primary Key or the Foreign Key fields. This allows the user the ability to insert new rows with empty values and fill in those values at a later point in time.

[0039] Other features may be implemented to help a user enter correct data into an XML document. When the XML is represented hierarchically, with tables in a relation being nested, inserting a row into the child table will automatically populate foreign key values from the primary key of the parent row. For the case where the tables are not nested, if the insert happens from a view that knows the context of the parent row (*e.g.*, the Master-Detail view) the Foreign-Key values may be populated automatically to match the Primary Key values of the current parent row.

[0040] Once a series of edits to the XML has been completed and the information is ready to be posted back to the database, the operations of **Fig. 7** are performed to identify changes to the data. The operations of Fig. 7 may be implemented by XML conversion module 230. At operation 710 the modified data is searched for duplicate annotated identifiers. In an exemplary implementation

the search may be performed by traversing the nodes of the modified data tree and searching the remaining nodes for duplicates of the annotated identifier. If, at operation 715, a duplicate annotated identifier is located, then each duplicate is checked to determine whether it exists at the original location where that identifier was assigned. If one is found in the original position then that node is selected and the annotated identifiers are removed from the remaining matching nodes. If none of the nodes with a duplicate identifier exist at the original location then at operation 725 the first node in the data with the given annotated identifier is selected, and the annotated identifiers from the remaining matching nodes are removed. By contrast, if at operation 715 no duplicate annotated identifiers are located, then at operation 725 the only node in the data with the given annotated identifier is selected.

[0041] At operation 735 the before copy of the data is searched for an annotated identifier matching the selected annotated identifier. If, at operation 740, a matching annotated identifier is found in the before copy of the data, then update procedures are invoked to update the corresponding data in the database.

[0042] In an exemplary implementation, the update procedures may be implemented as follows. First, if there is no database column associated with the selected data item, it is skipped. Second, if the associated database column represents an identity or a datestamp or timestamp, it is skipped. Third, if the associated database column is part of the relationship between this node and its

parent (*i.e.*, a foreign key), then the corresponding value from the parent node is used. Fourth, if the value is empty and the database field is optional, it is updated with a null value. By contrast, if the database field is required, it is updated with an empty string. Fifth, the new value from the selected data item is used to update the row's corresponding column.

[0043] By contrast, if at operation 740 there are no matching annotated identifiers, then insert procedures are invoked to write the edited data back to the database(s). In an exemplary implementation, the insert procedures may be implemented as follows. First, if there is no database column associated with the data item, the entry is skipped. Second, if the associated database column is an identity or datestamp or timestamp, then it is skipped. Third, if the associated database column is part of the relationship between this node and its parent (*i.e.*, a foreign key), then the corresponding value from the parent node is used. Fourth, if the value is empty and the database field is optional, then it is skipped. By contrast, if the database field is required, then it is inserted with an empty string. Fifth, the new value from the data item is used to populate the row's corresponding column.

[0044] The operations 735-750 may be repeated for each entry in the edited copy of the data to write the entries back to the database(s) from which the data was retrieved. At operation 755 delete procedures are invoked for any entries in the before copy of the data that were not processed as either an update or an insert.

The delete procedures may be implemented by deleting the row(s) corresponding to the entry (or entries) in the before copy of the data.

[0045] If the XML document is going to continue to be used for editing, then the XML document is updated to reflect the changes that have been made. In an exemplary implementation the following operations may be performed, *e.g.*, by the XML re-annotation module 232, to update the XML document. For each inserted row that contains an identity column (*i.e.*, auto-numbered) the generated identity value may be propagated back into the corresponding node and data item in the XML Document. For each inserted or updated row that contains a date/timestamp column, the new generated value may be propagated back into the corresponding node and data item in the XML document. For any row that had data items updated based upon the relationship with its parent, these new values may be propagated back into the corresponding data items in the XML document. The XML Document may be re-annotated with unique identifiers. And fifth, a new copy of the XML document is taken to become the new before data.

[0046] If the relationship between the querying and submit is not straight forward (such as when tracking changes to database data that is received via web services), the data may be marked as read only to prevent further modifications by the user. After the user refreshes the data explicitly, the lock on editing may be removed. This locking procedure ensures that the user does not work with potentially stale data.

Exemplary Operating Environment

[0047] The various components and functionality described herein are implemented with a number of individual computers. **Fig. 8** shows components of a typical example of a computer environment 800, including a computer, referred by to reference numeral 802. The computer 802 may be the same as or different from computer 102 of Fig. 1. The components shown in Fig. 8 are only examples, and are not intended to suggest any limitation as to the scope of the functionality of the invention; the invention is not necessarily dependent on the features shown in Fig. 8.

[0048] Generally, various different general purpose or special purpose computing system configurations can be used. Examples of well known computing systems, environments, and/or configurations that may be suitable for use with the invention include, but are not limited to, personal computers, server computers, hand-held or laptop devices, multiprocessor systems, microprocessor-based systems, set top boxes, programmable consumer electronics, network PCs, network-ready devices, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and the like.

[0049] The functionality of the computers is embodied in many cases by computer-executable instructions, such as software components, that are executed

by the computers. Generally, software components include routines, programs, objects, components, data structures, etc. that perform particular tasks or implement particular abstract data types. Tasks might also be performed by remote processing devices that are linked through a communications network. In a distributed computing environment, software components may be located in both local and remote computer storage media.

[0050] The instructions and/or software components are stored at different times in the various computer-readable media that are either part of the computer or that can be read by the computer. Programs are typically distributed, for example, on floppy disks, CD-ROMs, DVD, or some form of communication media such as a modulated signal. From there, they are installed or loaded into the secondary memory of a computer. At execution, they are loaded at least partially into the computer's primary electronic memory.

[0051] For purposes of illustration, programs and other executable program components such as the operating system are illustrated herein as discrete blocks, although it is recognized that such programs and components reside at various times in different storage components of the computer, and are executed by the data processor(s) of the computer.

[0052] With reference to Fig. 8, the components of computer 802 may include, but are not limited to, a processing unit 804, a system memory 806, and a system bus 808 that couples various system components including the system

memory to the processing unit 804. The system bus 808 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example, and not limitation, such architectures include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISAA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as the Mezzanine bus.

[0053] Computer 802 typically includes a variety of computer-readable media. Computer-readable media can be any available media that can be accessed by computer 802 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer-readable media may comprise computer storage media and communication media. "Computer storage media" includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules, or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computer 802. Communication media typically embodies computer-

readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism and includes any information delivery media. The term "modulated data signal" means a signal that has one or more if its characteristics set or changed in such a manner as to encode information in the signal. By way of example, and not limitation, communication media includes wired media such as a wired network or direct-wired connection and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of any of the above should also be included within the scope of computer readable media.

[0054] The system memory 806 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 810 and random access memory (RAM) 812. A basic input/output system 814 (BIOS), containing the basic routines that help to transfer information between elements within computer 802, such as during start-up, is typically stored in ROM 810. RAM 812 typically contains data and/or software components that are immediately accessible to and/or presently being operated on by processing unit 804. By way of example, and not limitation, Fig. 8 illustrates operating system 816, application programs 818, software components 820, and program data 822.

[0055] The computer 802 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, Fig. 8 illustrates a hard disk drive 824 that reads from or writes to non-removable,

nonvolatile magnetic media, a magnetic disk drive 826 that reads from or writes to a removable, nonvolatile magnetic disk 828, and an optical disk drive 830 that reads from or writes to a removable, nonvolatile optical disk 832 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 824 is typically connected to the system bus 808 through a non-removable memory interface such as data media interface 834, and magnetic disk drive 826 and optical disk drive 830 are typically connected to the system bus 808 by a removable memory interface.

[0056] The drives and their associated computer storage media discussed above and illustrated in Fig. 8 provide storage of computer-readable instructions, data structures, software components, and other data for computer 802. In Fig. 8, for example, hard disk drive 824 is illustrated as storing operating system 816', application programs 818', software components 820', and program data 822'. Note that these components can either be the same as or different from operating system 816, application programs 818, software components 820, and program data 822. Operating system 816', application programs 818', software components 820', and program data 822'are given different numbers here to illustrate that, at a minimum, they are different copies. A user may enter commands and information

into the computer 802 through input devices such as a keyboard 836, and pointing device (not shown), commonly referred to as a mouse, trackball, or touch pad. Other input devices may include source peripheral devices (such as a microphone 838 or camera 840 which provide streaming data), joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 802 through an input/output (I/O) interface 842 that is coupled to the system bus, but may be connected by other interface and bus structures, such as a parallel port, game port, or a universal serial bus (USB). A monitor 844 or other type of display device is also connected to the system bus 808 via an interface, such as a video adapter 846. In addition to the monitor 844, computers may also include other peripheral rendering devices (*e.g.*, speakers) and one or more printers which may be connected through the I/O interface 842.

[0057] The computer may operate in a networked environment using logical connections to one or more remote computers, such as a remote device 850. The remote device 850 may be a personal computer, a network-ready device, a server, a router, a network PC, a peer device or other common network node, and typically includes many or all of the elements described above relative to computer 802. The logical connections depicted in Fig. 8 include a local area network (LAN) 852 and a wide area network (WAN) 854. Although the WAN 854 shown in Fig. 8 is the Internet, the WAN 854 may also include other networks. Such networking

environments are commonplace in offices, enterprise-wide computer networks, intranets, and the like.

[0058] When used in a LAN networking environment, the computer 802 is connected to the LAN 852 through a network interface or adapter 856. When used in a WAN networking environment, the computer 802 typically includes a modem 858 or other means for establishing communications over the Internet 854. The modem 858, which may be internal or external, may be connected to the system bus 808 via the I/O interface 842, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 802, or portions thereof, may be stored in the remote device 850. By way of example, and not limitation, Fig. 8 illustrates remote software components 860 as residing on remote device 850. It will be appreciated that the network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Conclusion

[0059] Although the described arrangements and procedures have been described in language specific to structural features and/or methodological operations, it is to be understood that the subject matter defined in the appended claims is not necessarily limited to the specific features or operations described.

Rather, the specific features and operations are disclosed as preferred forms of implementing the claimed present subject matter.